EE-559 Project Report on Pattern Recognition

for a German-Credit-Risk Dataset from Kaggle.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to Prof Keith Jenkins and the TA Cao Yinwen for providing me with the requisite guidance in order to complete this project and solidify my understanding of the different techniques of classification involved in it.

student

Jitesh Chawla

USCID-7522739610

Email-jchawla@usc.edu

Date- May 2' 2017

Abstract

There are a lot of factors that determine whether a customer belongs to a Good or a Bad risk category such as Age, Job, Saving accounts or Duration for which the credit is required and a lot more. These customers are classified into 2 categories that is, whether the customer to whom the credit is lended will either prove to be a Good or a Bad credit Risk for the system. The German Credit risk was divided into two Datasets-"Training Set"(80%) and the "Testing Set"(20%). Even before this the most important task of the entire Classification was to Pre-process the dataset which included various measures like -treatment of missing values, outlier detection and removal, categorical data merging, feature dimensionality reduction and normalization were used[3]. This pre-processed data was then sent to the various classifier models for training purpose and in turn apply those models on the incoming testing dataset. On the basis of the results obtained including the "***Accuracy"*** and ***"F-1 score"*** we observed that the "***Logistic Regression Model"*** gave the best F-1 score of 0.715 on the incoming testing data. From the result of ***Principal Component Analysis*** it was found that features like ***"Age", "Job", "Saving accounts" and "Duration"*** were amongst the key factors in terms of determining whether the particular customer was worth a good credit risk or bad. Several other features like the ***"Purpose", "Housing"*** and ***"Credit amount"*** also played a pivotal role in classifying the customers appropriately.[11][3]

Goals

The primary goal of this project is to develop a Pattern Recognition System that operates on German Credit Risk Dataset to appropriately determine if the customer belonged to a ***"Good-Credit Risk" or*** "***Bad-Credit Risk***" class on the basis of Overall Accuracy greater than 70%. It was also desirable to determine the features which were most important in training the model and contributed in prediction of the classes for the testing dataset. In order to implement this system the Kaggle Dataset had to be thoroughly cleaned and pre-processed before it was sent to various Classifiers in order to model them for the Testing Dataset.

Pre-processing of DataSet

After the original dataset was divided in two sets with the (80-20) composition as the Training Set and the Testing Set. The following procedure is outlined below.[3][4]

1. Splitting into Training and Testing Set .

The most important library for Machine learning i.e **sklearn** was installed in python. By using its inbuilt **train_test_split()** function the German Credit Risk Dataset was divided into Training Set(***data_train, data_test)*** and correspondingly Testing Set (***target_train*** and ***target_test).***[8]
The preprocessing is applied on the training Data set for designing the Pattern recognition System. The testing dataset will be totally kept aside until the classifier is trained and then that classifier will be applied to the testing data set.[1]

2. Actual Preprocessing-

The Dataset(.csv) was converted to a ***Pandas Data frame*** named as the Utility_ Matrix1 by installing the pandas library in python. Following techniques outlined below were used to perform the pre-processing on the dataset.[3][9]

A. Feature Recasting/Conversion-

After extracting the contents of a csv file to a Pandas Data frame the foremost important task is to convert the entire dataset into numeric datatype, whereas already numerical feature were left unchanged. For features like Checking and Saving Accounts where we could assign a sequential oder to

the various categories, we did assign in a particular order by using the ***replace()*** function of the data frame and assigned the following values as:
"***little" -> 0 , "moderate" -> 1 , "rich" -> 2 , "quite rich" -> 3.*** [4]

B. Data Point Reduction-

This part of preprocessing is really important as there is no point considering those features which do not have enough information so that they can contribute enough to the training of the classifier.
The missing data in the German Credit Risk data set got converted to "***NaN***" values and this happened only for the ***"Saving account"*** and ***"Checking Account"*** features as those were the only two features having missing values.
By using the ***value_count()*** we observed that almost on-third entries were missing for the Checking Account feature. This wouldn't have contributed much to the training so we decided to drop it.[7]

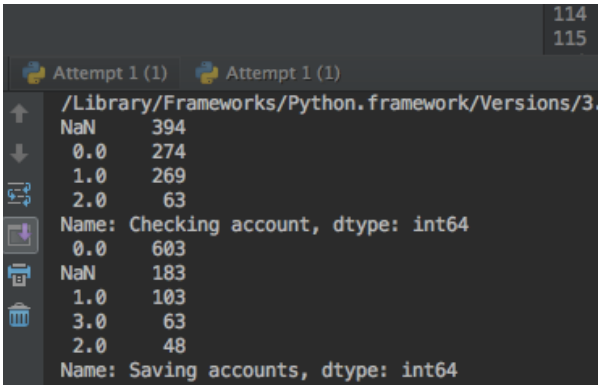


figure1- Count of Nan values for both the features

C. Imputing Missing Values-

Similarly, for the the Saving accounts feature there were far less missing values which were imputed with the mean values of the entire column by creating a ***DataFrame Imputer Class*** by importing the ***Transformermixin*** module from the ***sklearn,base*** package.[1]

D. Categorical Feature handling-

This section of preprocessing dealt with features such as ***"Sex", "Housing" and "Purpose"*** containing string values which needed to be encoded in the form of binary vectors. According to this conversion, for example, the feature Housing got split into -> "***Housing_own***", "***Housing_free***", "***Housing_Rent"*** and similarly for Sex and Purpose features too.[3]
Attached below are a few screenshots of the output console screen just to present a feel of the data after few steps of preprocessing.[2]

Sex_female	Sex_male
0	1
1	0
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1

figure2- Binary vector for the Sex Feature

Housing_free	Housing_own	Housing_rent
0	1	0
0	1	0
0	1	0
1	0	0
1	0	0
1	0	0
0	1	0
0	0	1

figure2- Binary vector for the Housing Feature

Purpose_furniture/equipment	Purpose_radio/TV	Purpose_repairs
0	1	0
0	1	0
0	0	0
1	0	0
0	0	0
0	0	0

Purpose_business	Purpose_car	Purpose_domestic appliances	Purpose_education
0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0
0	1	0	0

figure4- Binary vector for the Purpose Feature

E. Feature Rescaling-

Passing the data to the ***StandardScaler()*** function imported from the ***sklearn.preprocessing*** package of the sklearn library in python. By default all the values are rescaled between (0,1)

F. Dimensionality Reduction-

There were many features in the German Credit Risk Dataset that were not significant enough and did not contain any relevant information to be considered towards training of the classifier. These features would have skewed the classification results.
This was achieved by performing ***Principal Component Analysis*** and ***SVD(Single Valued Decomposition)***

In PCA, first of all the dataset was passed through a ***StandardScaler()*** function then the corresponding values for the ***Covariance Matrix*** via ***numpy.cov()*** function are calculated.[12]
Also, the corresponding ***eigen_values*** and ***eigen_vectors*** are calculated via ***np.linalg.eig(cov_mat)*** function.[4]

After calculating all the eigen values of the corresponding features, they were sorted in a descending order and the first 14 values were taken into consideration and ignoring the rest as their eigen values were very low.

We were left with a reduced feature space and in order to cross check our results whether they were actually correct, we performed another technique that is the ***SVD***.
It was implemented by the help of numpy library function i.e ***numpy.linalg.svd()*** function. Much to our expectation the results were the same for both the techniques and they had been confirmed.

Classifiers

There were several Classifiers used in this project such as the ***"Logistic Regression", "Ada Boosting", "Random Forest", "KNN Model", "Naive Bayes Classifier"*** and ***"SVM" .***
All the above models were trained on the training dataset, and then applied to the testing set which was left uncorrupted in the beginning of the analysis. On the basis of the Classification_report() parameters like the precision, recall, F-1 score and support along with the accuracy of the predictions, ***Logistic Regression model*** gave the best results.

Along with the Logistic Regression Model Naive Bayes Clasifier and Random Forest Classifiers were chosen and our described briefly below.

Logistic Regression Model

This classifier was implemented by directly using an inbuilt function of ***sklearn.linear_model*** package in python called the ***LogisticRegression().***
logreg.fit(data_train,target_train) is used to fit the model on the training data and with the help of ***logreg.predict(data_test)*** we get the predicted values for our testing dataset. As we know, it is a regression model where the dependent variable (DV) is categorical. It usually covers the case of a binary dependent variable—that is, where it can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/ lose, alive/dead or healthy/sick. In our project it meant either ***Good credit/Bad Credit Risk.***

Several Validation techniques like ***Data-splitting*** in which the original sample is randomly split into the fitting and validation samples and ***Iterations over the full model*** were taken into account before validating our Classifier.

Cross validation Scores were also calculated for a ***5 fold cross validation*** by using the inbuilt ***cross_val_score()*** from sklearn Library.[5][6]
The ***F-1-0.67*** and ***Accuracy-0.715*** was recorded for the ***Logistic Regression Model.***

Naive Bayes Model -

This classifier was implemented by directly using an inbuilt function of ***sklearn.naive_bayes*** package in python called the ***GaussianNB().***
gnb.fit(data_train,target_train) is used to fit the model on the training data and with the help of ***gnb.predict(data_test)*** we get the predicted values for our testing dataset. As we know, Naive babes is not a distribution free classifier, it requires the class priors and is based on the concept of ***PDF(Probabiiity density Function).***
Several distribution methods such as the ***"Normal/Gaussian", "Kernel", "Multinomial "*** were tried and we go the best results in the case of a Normal Distribution. Best accuracy and F-1 scores were recorded for the ***Normal distribution*** amongst the other distributions used for the Naive Bayes Classifier.
Cross validation Scores were also calculated for a ***5 fold cross validation*** by using the inbuilt ***cross_val_score()*** from sklearn Library.
The ***F-1-0.66*** and ***Accuracy-0.665*** was recorded for the Naive Bayes Model.[6]

Random Forest Model-

This classifier was implemented by directly using an inbuilt function from ***sklearn.ensemble*** package import python called the ***RandomForestClassifier().***
rnd_fc.fit(data_train,target_train) is used to fit the model on the training data and with the help of ***rnd_fc.predict(data_test)*** we get the predicted values for our testing dataset. As we know, **Random forests** or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Cross validation Scores were also calculated for a 5 fold cross validation by using the inbuilt cross_val_score() from sklearn Library.
The ***F-1-0.66*** and ***Accuracy-0.695*** was recorded for the ***Random Forest Model***.[5]

Performance Evaluation Techniques-

We calculated the performance of our various classifiers by passing the Testing dataset through the exact same process as the training Set.
First and most important step was Preprocessing of the testing set very much like the training set. Then this preprocessed test set was allowed to pass through various Classifiers trained earlier.

By using an inbuilt function of ***sklearn.metrics*** library in python we used the ***classification_report()*** function to get the various resulting parameters for comparison.

Also using the same library we used another function to calculate the ***Confusion_Matrix*** for the testing dataset in all the three classifiers to compare with each other. it gave the performance of each Model on testing set whose true value is known.

Results were plotted graphically as well to make it look more comprehensive and legible by using the ***matplotlib.pyplot*** library in python.

Results-

The results of various Classifiers used during the Analysis of the German Credit Risk Dataset are shown below by using the ***c l a s s i f i c a t i o n _ r e p o r t ( )*** f r o m ***skleran.metrics*** library in python.[5][6]

==========

Logistic regression - Best Model

Cross_validation_scores = [0.695 0.685 0.72 0.71 0.74]
Accuracy: 0.71 (+/- 0.04)
0.715

	precision	recall	f1-score	support
class 1	0.74	0.92	0.82	143
class 2	0.50	0.19	0.28	57
avg / total	0.67	0.71	0.67	200

==========

Random Forest

Cross_validation_scores = [0.675 0.675 0.68 0.685 0.71]
Accuracy: 0.69 (+/- 0.03)
0.695

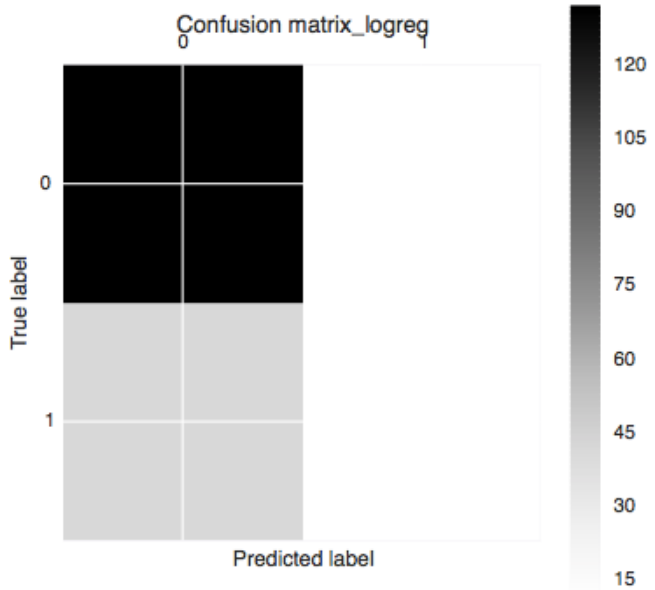
	precision	recall	f1-score	support
class 1	0.74	0.89	0.81	143
class 2	0.43	0.21	0.28	57
avg / total	0.65	0.69	0.66	200

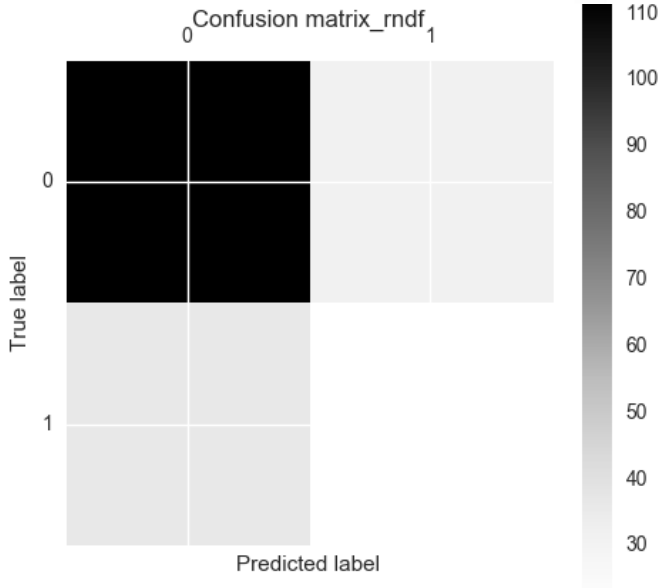
==========

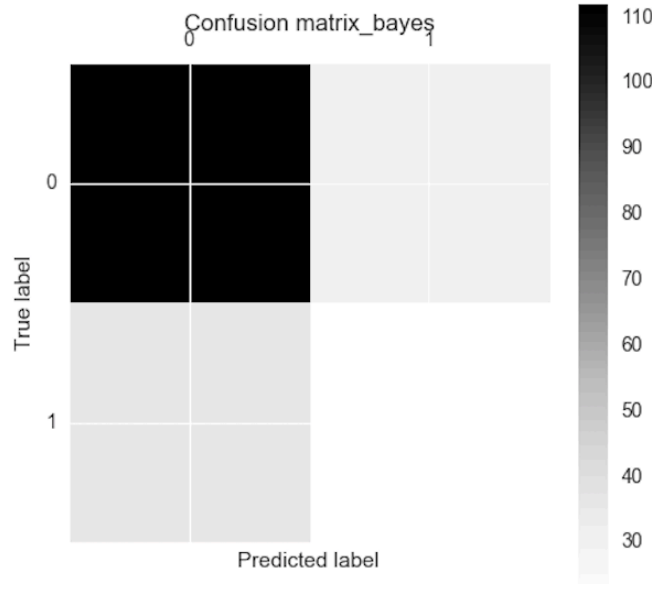
Bayes Minimum Classifier

Cross_validation_scores = [0.68 0.64 0.66 0.7 0.675]
Accuracy: 0.67 (+/- 0.04)
0.665

	precision	recall	f1-score	support
class 1	0.76	0.78	0.77	143
class 2	0.40	0.37	0.39	57
avg / total	0.66	0.67	0.66	200







Interpretation / Conclusions-

3. We realized that analysis of the dataset is the most important part of the entire project on improving the classification report parameters like the F-1 score , accuracy , precision and recall. It was evident from the fact that there were a few features that did not contribute much to the model training and were thus discarded from the dataset.
4. Important features that were responsible for deciding whether a customer belongs to a Good credit risk or Bad are ***Age", "Job", "Saving accounts" and "Duration".***
5. The results of the logistic regression model were the best amongst all the classifiers that were implemented in terms of F-1 score and the Accuracy obtained.
6. Principal Component Analysis played a pivotal role too in determining the key features for the Credit Risk Data Set and the results were conformed with those of SVD.
7. The "***Checking Account"*** feature had a large number of missing values and instead of imputing them it was beneficial to eliminate that feature from the dataset for the training purpose.
8. The ***"Credit Amount "*** had a really high value of variance as compared to all other features in the dataset.
9. Feature Expansion and Encoding the categorical data for features like ***"Hosuing", "Purpose"*** and ***"Sex"*** definitely played a big role in boosting up the accuracy and the F-1 score.
10. Also, by finding Cross-validation-values by simply using ***Cross_val_score()*** from sklearn.model_selection package of sk learn in python we could validate our model.

References

1. **SUPERVISED LEARNING — SCIKIT-LEARN 0.18.1 DOCUMENT**
   In-text: [8]
   Your Bibliography: [8]"1. Supervised learning — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/supervised_learning.html#supervised-learning. [Accessed: 23- Apr- 2017].

2. **DECOMPOSING SIGNALS IN COMPONENTS (MATRIX FACTORIZATION PROBLEMS) — SCIKIT-LEARN 0.18.1**
   In-text: [7]
   Your Bibliography: [7]"2.5. Decomposing signals in components (matrix factorization problems) — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/decomposition.html#decompositions. [Accessed: 24- Apr- 2017].

3. **PREPROCESSING DATA — SCIKIT-LEARN 0.18.1**
   In-text: [3]
   Your Bibliography: [3]"4.3. Preprocessing data — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing. [Accessed: 26- Apr- 2017].

4. **ANON**
   In-text: [10]
   Your Bibliography: [10]2017. [Online]. Available: https://cseweb.ucsd.edu/~akmenon/ResearchExam.pdf. [Accessed: 25- Apr- 2017].

5. **GERMAN CREDIT RISK I KAGGLE**
   In-text: [11]
   Your Bibliography: [11]"German

Credit Risk | Kaggle", *Kaggle.com*, 2017. [Online]. Available: https://www.kaggle.com/uciml/german-credit. [Accessed: 26- Apr- 2017].

6. **IMPLEMENTING A PRINCIPAL COMPONENT ANALYSIS (PCA)**
In-text: [4]
Your Bibliography: [4]"Implementing a Principal Component Analysis (PCA)", *Sebastian Raschka's Website*, 2017. [Online]. Available: http://sebastianraschka.com/Articles/2014_pca_step_by_step.html. [Accessed: 27- Apr- 2017].

7. **NUMPY — NUMPY**
In-text: [12]
Your Bibliography: [12]"NumPy — NumPy", *Numpy.org*, 2017. [Online]. Available: http://www.numpy.org. [Accessed: 20- Apr- 2017].

8. **PANDAS: POWERFUL PYTHON DATA ANALYSIS TOOLKIT — PANDAS 0.19.2**
In-text: [9]
Your Bibliography: [9]"pandas: powerful Python data analysis toolkit — pandas 0.19.2 documentation", *Pandas.pydata.org*, 2017. [Online]. Available: http://pandas.pydata.org/pandas-docs/stable/index.html. [Accessed: 22- Apr- 2017].

9. **SCIKIT-LEARN, I.**
Impute categorical missing values in scikit-learn
In-text: [1]
Your Bibliography: [1]I. scikit-learn, "Impute categorical missing values in scikit-learn", *Stackoverflow.com*, 2017. [Online]. Available: http://stackoverflow.com/questions/25239958/impute-categorical-missing-values-in-scikit-learn. [Accessed: 01- May- 2017].

10. **SKLEARN.METRICS.CLASSIFICATION_REPORT — SCIKIT-LEARN 0.18.1 DOCUMENTATION**
In-text: [5]
Your Bibliography: [5]"sklearn.metrics.classification_report — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html. [Accessed: 30- Apr- 2017].

11. **SKLEARN.MODEL_SELECTION.CROSS_VAL_SCORE — SCIKIT-LEARN 0.18.1**
In-text: [6]
Your Bibliography: [6]"sklearn.model_selection.cross_val_score — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html. [Accessed: 29- Apr- 2017].

12. **Converting categorical data** into numbers with Pandas and Scikit-learn - FastML
In-text: [2]
Your Bibliography: [2]Z. Z., "Converting categorical data into numbers with Pandas and Scikit-learn - FastML", *Fastml.com*, 2017. [Online]. Available: http://fastml.com/converting-categorical-data-into-numbers-with-pandas-and-scikit-learn/. [Accessed: 25- Apr- 2017].